Aufgabe: Sie befinden sich im Punkt 1 und wollen der Reihe nach $n-1$ Besichtigungspunkte aufsuchen. In welcher Abfolge müssen Sie diese Punkte anfahren, um das Besichtigungsprogramm so schnell wie möglich zu absolvieren, wenn Ihr letzter Besichtigungspunkt der Punkt n sein soll? Berechnen Sie im konkreten Fall gemäß nachfolgender Tabelle den kürzesten Weg im Falle 6 disjunkter Wegpunkte (in einem kartesischem Koordinatensystem) sowie für die geschlossene Trajektorie.

i	x_i	y_i	i	x_i	y_i
1	0	0	1	0	0
2	5	15	2	5	15
3	8	3	3	8	3
4	11	21	4	11	21
5	15	9	5	15	9
6	10	10	6	0	0

Hinweis: Nehmen Sie als Fahrtstrecke vereinfachend geradlinige Verbindungen zwischen je zwei Punkten an sowie eine konstante Fahrtgeschwindigkeit.

Lösung: Wir numerieren die Punkte der Reihe nach wie oben angegeben mit natürlichen Zahlen, wobei es auf die Reihenfolge und den Abstand zwischen je zwei Punkten zunächst nicht ankommt. Sodann berechnen wir für alle möglichen Permutationen dieser Wegpunktfolge die Gesamtlängen

$$s_{1ijkl\ldots n} = \sqrt{(x_i - x_1)^2 + (y_i - y_1)^2} + \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} + \sqrt{(x_k - x_j)^2 + (y_k - y_j)^2} + \sqrt{(x_l - x_k)^2 + (y_l - y_k)^2} + \sqrt{(x_\xi - x_l)^2 + (y_\xi - y_l)^2} + \ldots + \sqrt{(x_n - x_\zeta)^2 + (y_n - y_\zeta)^2}$$

und ermitteln deren Minimum, welches uns die kürzeste Wegstrecke liefert. Im konkreten Fall von 6 Wegpunkten, wobei Anfang und Ende fest vorgegeben sind, haben wir 4! = 24 Permutationen

s_{123456}	s_{123546}	s_{124356}	s_{124536}	s_{125346}	s_{125436}
s_{132456}	s_{132546}	s_{134256}	s_{134526}	s_{135246}	s_{135426}
s_{142356}	s_{142536}	s_{143256}	s_{143526}	s_{145236}	s_{145326}
s_{152346}	s_{152346}	s_{152346}	s_{152346}	s_{152346}	s_{152346}

auf ihren Minimalwert zu überprüfen. Dazu berechnen wir gemäß Tabelle 1 alle möglichen Kombinationen

$$s_{1ijkl6} = \sqrt{(x_i - x_1)^2 + (y_i - y_1)^2} + \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} + \sqrt{(x_k - x_j)^2 + (y_k - y_j)^2} + \sqrt{(x_l - x_k)^2 + (y_l - y_k)^2} + \sqrt{(x_6 - x_l)^2 + (y_6 - y_l)^2}.$$

Das Ergebnis ist in Abbildung 1 und Abbildung 2 dargestellt. Im Falle der offenen Trajektorie besitzt die Permutation s_{135426} die kürzeste Weglänge, im Falle der geschlossenen ist die Lö-

sung nicht eindeutig, weil es je nach Durchlaufrichtung zwei Permutationen gibt, die gleiche Weglänge haben: s_{124536} und s_{135426}. In der Praxis wird man daher noch ein zweites Kriterium hinzuziehen, um sich für eine der beiden Varianten zu entscheiden.

i	j	k	l	s_{1ijkl6}	i	j	k	l	s_{1ijkl6}
2	3	4	5	64,1771	2	3	4	5	76,5710
2	3	5	4	61,0947	2	3	5	4	73,7559
2	4	3	5	56,8635	2	4	3	5	69,2574
2	4	5	3	53,4454	2	4	5	3	54,7093
2	5	3	4	65,9865	2	5	3	4	78,6477
2	5	4	3	65,6508	2	5	4	3	66,9147
3	2	4	5	47,1467	3	2	4	5	59,5406
3	2	5	4	56,2697	3	2	5	4	68,9309
3	4	2	5	52,0385	3	4	2	5	64,4323
3	4	5	2	58,1744	3	4	5	2	66,9147
3	5	2	4	48,9561	3	5	2	4	61,6173
3	5	4	2	45,9690	3	5	4	2	54,7093
4	2	3	5	58,8797	4	2	3	5	71,2735
4	2	5	3	60,3534	4	2	5	3	61,6173
4	3	2	5	71,0851	4	3	2	5	83,4789
4	3	5	2	69,9073	4	3	5	2	78,6477
4	5	2	3	67,6670	4	5	2	3	68,9309
4	5	3	2	65,0156	4	5	3	2	73,7559
5	2	3	4	70,8177	5	2	3	4	83,4789
5	2	4	3	63,1684	5	2	4	3	64,4323
5	3	2	4	58,6124	5	3	2	4	71,2735
5	3	4	2	60,5170	5	3	4	2	69,2574
5	4	2	3	58,2767	5	4	2	3	59,5406
5	4	3	2	67,8306	5	4	3	2	76,5710

Tabelle 1. Permutationsweglängen zwischen Punkt 1 und Punkt 6 (links) sowie für identischen Anfangs- und Endpunkt (rechts)

Im Anhang ist ein Algorithmus angegeben, mit dem man die genannten Berechnungen durchführen kann. Wenn man diesen auf mehr als 6 Punkte erweitern will, muß man für jede Erweiterung eine zusätzliche Schleife einbauen. Zur Minimierung von n Wegpunkten müssen also insgesamt $(n-2)!$ Permutationen durchlaufen werden, was die Methode natürlich begrenzt.

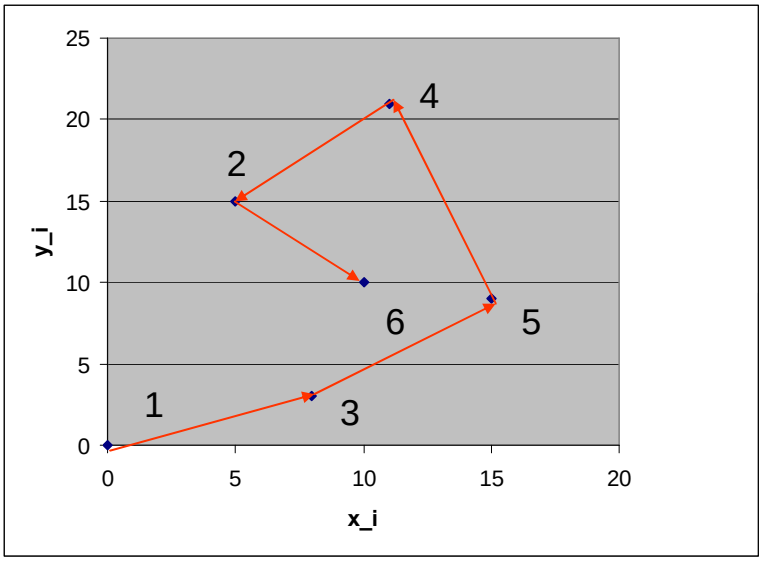

Abbildung 1. Minimale Strecke zwischen Punkt 1 und Punkt 6

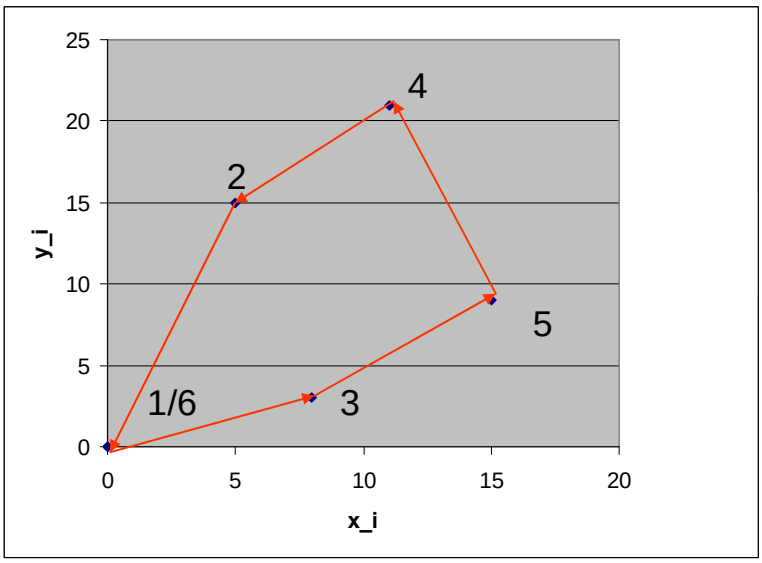

Abbildung 2. Geschlossene Minimalweglänge mit identischem Anfangs- und Endpunkt

Diese Aufgabe ist grundlegend für die autonome Wegfindung, wenn als einziges Kriterium die Spritersparnis zugrunde liegt. Das autonome System hat die freie Auswahl zwischen 24 alternativen Möglichkeiten und findet daraus selbständig die beste Lösung. Was aber die beste Lösung ist, muß anhand von Maßgaben vorgegeben werden, damit Zufall und Willkür ausgeschlossen bleiben. Freiheit kann sich also stets nur innerhalb eines vorgegebenen Wertesystems bewegen.

Anhang

```
% Permutationen

x(1) = 0;  y(1) = 0;
x(2) = 5;  y(2) = 15;
x(3) = 8;  y(3) = 3;
x(4) = 11; y(4) = 21;
x(5) = 15; y(5) = 9;
x(6) = 10; y(6) = 10;

s1 = sqrt((x(2) - x(1))^2 + (y(2) - y(1))^2);
s2 = sqrt((x(3) - x(2))^2 + (y(3) - y(2))^2);
s3 = sqrt((x(4) - x(3))^2 + (y(4) - y(3))^2);
s4 = sqrt((x(5) - x(4))^2 + (y(5) - y(4))^2);
s5 = sqrt((x(6) - x(5))^2 + (y(6) - y(5))^2);

smin = s1 + s2 + s3 + s4 + s5;
imin = 2;
jmin = 3;
kmin = 4;
lmin = 5;

for i = 2:5
   s1 = sqrt((x(i) - x(1))^2 + (y(i) - y(1))^2);
   for j = 2:5
      if j ~= i
         s2 = sqrt((x(j) - x(i))^2 + (y(j) - y(i))^2);
         for k = 2:5
            if (k ~= j) & (k ~= i)
               s3 = sqrt((x(k) - x(j))^2 + (y(k) - y(j))^2);
               for l = 2:5
                  if (l ~= k) & (l ~= j) & (l ~= i)
                     s4 = sqrt((x(l) - x(k))^2 + (y(l) - y(k))^2);
                     s5 = sqrt((x(6) - x(l))^2 + (y(6) - y(l))^2);
                     s = s1 + s2 + s3 + s4 + s5;
                     if s < smin
                        smin = s;
                        imin = i;
                        jmin = j;
                        kmin = k;
                        lmin = l;
                     end;
                     vec = [int8(i), int8(j), int8(k), int8(l)], s
                  end;
               end;
            end;
         end;
      end;
   end;
```



```
end

vec = [int8(imin), int8(jmin), int8(kmin), int8(lmin)], smin
```

